Ex. No: 08	Creation Of Other Database Objects
Date	10/10/2023

Objective

To create other database objects like views, index, synonyms etc.

Description

A database object is any defined object in a database that is used to store or reference data. Anything which we make from create command is known as Database Object. It can be used to hold and manipulate the data. Some of the examples of database objects are: view, sequence, indexes, etc.

- Table – Basic unit of storage; composed rows and columns
- View – Logically represents subsets of data from one or more tables
- Sequence – Generates primary key values
- Index – Improves the performance of some queries
- Synonym – Alternative name for an object

View – This database object is used to create a view in database.A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables. The view is stored as a SELECT statement in the data dictionary.

Syntax :

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
        [(alias [, alias]...)]
        AS subquery
        [WITH CHECK OPTION [CONSTRAINT constraint]]
        [WITH READ ONLY [CONSTRAINT constraint]];
```

Sequence – This database object is used to create a sequence in database.A sequence is a user created database object that can be shared by multiple users to generate unique integers. A typical usage for sequences is to create a primary key value, which must be unique for each row.The sequence is generated and incremented (or decremented) by an internal Oracle routine.

Syntax :

```
CREATE SEQUENCE sequence
        [INCREMENT BY n]
        [START WITH n]
        [{MAXVALUE n | NOMAXVALUE}]
        [{MINVALUE n | NOMINVALUE}]
        [{CYCLE | NOCYCLE}]
        [{CACHE n | NOCACHE}];
```

Index – This database object is used to create a indexes in database.An Oracle server index is a schema object that can speed up the retrieval of rows by using a pointer.Indexes can be created explicitly or automatically. If you do not have an index on the column, then a full table scan occurs.

An index provides direct and fast access to rows in a table. Its purpose is to reduce the necessity of disk I/O by using an indexed path to locate data quickly. The index is used and maintained automatically by the Oracle server. Once an index is created, no direct activity is required by the user.Indexes are logically and physically independent of the table they index. This means that they can be created or dropped at any time and have no effect on the base tables or other indexes.

Syntax :

```
CREATE INDEX index
        ON table (column[, column]...);
```

Synonym – This database object is used to create a indexes in database.It simplify access to objects by creating a synonym(another name for an object). With synonyms, you can Ease referring to a table owned by another user and shorten lengthy object names.To refer to a table owned by another user, you need to prefix the table name with the name of the user who created it followed by a period. Creating a synonym eliminates the need to qualify the object name with the schema and provides you with an alternative name for a table, view, sequence,procedure, or other objects. This method can be especially useful with lengthy object names, such as views.

In the syntax:

PUBLIC : creates a synonym accessible to all users

synonym : is the name of the synonym to be created

object : identifies the object for which the synonym is created

Syntax :

CREATE [PUBLIC] SYNONYM synonym FOR object;

Questions

1. Design a view that present a list of venues along with the cities.

```
SQL> CREATE OR REPLACE VIEW VenueCityList AS
  2  SELECT VenueID, Name AS VenueName, City
  3  FROM Venue_1128;

View created.
```

2. Create a view that combines data from the user and Events tables to display the name of each user along with the event names.

```
SQL> CREATE OR REPLACE VIEW UserEventList AS
  2  SELECT U.Name AS UserName, E.Name AS EventName
  3  FROM User_1128 U
  4  JOIN Ticket_1128 T ON U.UserID = T.UserID
  5  JOIN Event_1128 E ON T.EventID = E.EventID;

View created.
```

3. Build a view that shows a summary of the number of events in each venue.

```
SQL> CREATE OR REPLACE VIEW VenueEventSummary AS
  2  SELECT V.VenueID, V.Name AS VenueName, COUNT(E.EventID) AS EventCount
  3  FROM Venue_1128 V
  4  LEFT JOIN Event_1128 E ON V.VenueID = E.VenueID
  5  GROUP BY V.VenueID, V.Name;

View created.
```

4. Display the 3 oldest users from the users table.

```
SQL> select userid,name
  2  from(select userid,name
  3  from User_1128
  4  order by userid)
  5  where rownum <= 3;

    USERID
----------
NAME
----------------------------------------
         1
John Smith

         2
Jane Doe

         3
Michael Lee
```

5. Display the last 5 events from the events table.

```
SQL> select Eventid,Name from(select EventID,Name from Event_1128
  2  order by EventDate desc) where rownum <= 5;

   EVENTID
----------
NAME
----------------------------------------------------------------------
         8
Dance Workshop

         7
Tech Conference

         6
Comedy Show


   EVENTID
----------
NAME
----------------------------------------------------------------------
         5
Food Festival

         4
Art Exhibition
```

6. Create a sequence that generates unique user IDs starting from 1001 and incrementing by 1 for each new user added to the users table. Add 3 new records using the sequence.

```
SQL> CREATE SEQUENCE UserID_Seq
  2    START WITH 1001
  3    INCREMENT BY 1
  4    NOCACHE;

Sequence created.
```

```
SQL> -- Insert the first new user
SQL> INSERT INTO User_1128 (UserID, Name, Email, Password, Phone)
  2  VALUES (UserID_Seq.NEXTVAL, 'John Doe', 'john@example.com', 'password123', '123-456-7890');

1 row created.

SQL>
SQL> -- Insert the second new user
SQL> INSERT INTO User_1128 (UserID, Name, Email, Password, Phone)
  2  VALUES (UserID_Seq.NEXTVAL, 'Jane Smith', 'jane@example.com', 'password456', '987-654-3210');

1 row created.

SQL>
SQL> -- Insert the third new user
SQL> INSERT INTO User_1128 (UserID, Name, Email, Password, Phone)
  2  VALUES (UserID_Seq.NEXTVAL, 'Bob Johnson', 'bob@example.com', 'password789', '555-555-5555');

1 row created.
```

7. Display the current value of the sequence.

```
SQL> SELECT UserID_Seq.CURRVAL
  2  FROM dual;

   CURRVAL
----------
      1003
```

8. Alter the sequence to increment by 10. Add 3 records to the users table using the sequence.

```
SQL> ALTER SEQUENCE UserID_Seq INCREMENT BY 10;

Sequence altered.

SQL> -- Insert the first new user
SQL> INSERT INTO User_1128 (UserID, Name, Email, Password, Phone)
  2  VALUES (UserID_Seq.NEXTVAL, 'Alice Johnson', 'alice@example.com', 'password123', '123-456-7890');

1 row created.

SQL>
SQL> -- Insert the second new user
SQL> INSERT INTO User_1128 (UserID, Name, Email, Password, Phone)
  2  VALUES (UserID_Seq.NEXTVAL, 'Bob Smith', 'bob@example.com', 'password456', '987-654-3210');

1 row created.

SQL>
SQL> -- Insert the third new user
SQL> INSERT INTO User_1128 (UserID, Name, Email, Password, Phone)
  2  VALUES (UserID_Seq.NEXTVAL, 'Charlie Brown', 'charlie@example.com', 'password789', '555-555-5555');

1 row created.
```

9. Create an index on the userid column of the users table and check the access time with userid in the where clause.

```
SQL> create index id on User_1128(userid);
```

```
Index created.
```

10. Create a synonym for the users table.

```
SQL> CREATE SYNONYM UserSynonym FOR User_1128;

Synonym created.
```

Result:

Thus the queries for creating other database objects are done successfully.